タッチ操作向け画像資料の電子付箋基盤のデザインとその実装

A Design and Implementation of the Digital Annotation Basis on an Image Resource for a Touch Operation

津田光弘

TSUDA Mitsuhiro

[論文要旨]

本稿は，歴史資料のデジタル画像について，その画像上の文字位置や行を電子書籍のテキストのように選択可能にするユーザーインタフェースのデザインと実装について記載している。これはタブレットデバイスによる大量の電子付箋の作成基盤の準備を意識している。

先ず，電子付箋と電子書籍アプリケーションのマーカツールの分析，そして資料の文字レイアウト構造の特徴から，画像文字の暗黙的かつ基本的な性質である「流れ」すなわち「テキスト動線」の概念を得た。これは画像上の文字列のレイアウトとテキストデータをつなぐ概念である。

次に，テキスト動線の概念から，擬似的かつ仮想的なテキストデータ列である電子付箋の基盤 Vein（ベイン）を設計した。この電子付箋基盤は以下の特徴を持つ。(1) タッチ操作を受け付ける拘束的な道筋を準備する，(2) 自らをアノテーションを生成する型とする，(3) 縦書きや横書きなどの各言語の手書き文字のレイアウトに対応できる，(4) HTML5 の SVG を用いて様々な行レイアウトに対応できる，(5) 複数行を続けて選択できる，(6) 手作業で作成可能なように簡単なデータ構造を持つ。

最後に，一般化データモデルの設計を経て，以上のデザインモデルを HTML5 と CSS3, JavaScript によって実装した。そして，タッチ操作によるタブレットデバイスと従来のマウス操作によるパーソナルコンピュータ両方の Web ブラウザ上で動作を検証し，このユーザーインタフェースの有用性を確認すると共に，課題について考察した。本稿の付録として主たる JavaScript プログラムを付し，テキスト動線データの製作ツールおよび動作サンプルを記載した著者の Web サイトを案内している。

【キーワード】画像，電子付箋，アノテーション，タッチデバイス，ユーザーインタフェース

❶……はじめに

デジタル化された歴史研究情報として，これまで数多く製作され蓄積されている古い書物や資料の画像を考えるとき，その分析利用を目的として電子付箋あるいはアノテーションと呼ばれる手法がある。この手法はデジタル画像の画素座標に基づき四角形や円などの図形を描き，その図形状の付箋に対しメタデータを登録する。これは付箋紙の考え方をデジタル化したものである。元の画像に影響せず，個人やグループでの検討やデータベースにも応用できるが，マウスなどのポインティングデバイスを用いた精緻な作業が要求されやすく，文字を主体とした画像資料よりも古地図や絵巻などの描画箇所の分析に用いられることが多い。

一方，約 30 年間に渡りコンピュータ操作のユーザーインタフェースとしてマウス等のポインティングデバイスが主流であったが，ここ数年間でタッチ操作と表示が一体化したタブレットデバイスが急速に普及している。このデバイス用のアプリケーションとして先行した電子書籍では「しおり（ブックマーク）」「マーカ」といった電子付箋に類する機能が提供されており，テキストデータの表示文字列に対しタッチ操作でラインマーカを引くことができる。この手順は図形の場合に比べて簡易であり，行を折り返す文字資料の構造にも適合している。

歴史研究の画像資料には文字を主とするものが多いが，これらをタブレットデバイスで扱う場合に，電子書籍アプリケーションで例示したようなタッチ操作でラインマーカが作成できれば，図形状の電子付箋より気軽に利用ができ，画像資料の詳細分析やリンク作成などに役立つと考える。しかし，現状では歴史資料において，画像の状態にある文字（以下，画像文字とする）をテキストデータ化するには技術上の困難が伴い，電子書籍のようなユーザーインタフェース体験を利用者にあまねく提供できない。そこで，画像中にテキストデータの構造の代わりとなる仕組みを検討し，画像文字向けの新たな電子付箋とその基盤をデザインした。

本稿では，国立歴史民俗博物館基盤研究「デジタル化された歴史研究情報の高度利用に関する研究」（代表：鈴木卓治，平成 22 ～ 24 年度）において，タッチ操作での利用を主に想定した電子付箋を構想し，モデル化，ユーザーインタフェースのデザインを経て，プログラムコードとして実装した成果およびプロトタイプの操作評価について報告する。

なお，本稿中，アノテーション，マーカなどツールの種類によって様々な呼び方があるが混乱を招きやすい。以下では特定する場合を除き「電子付箋」と記載する。

❷……電子付箋の再考

電子付箋は付箋紙のデジタル表現であるが，現実の付箋紙に様々な種類があるように，電子付箋にも用途に応じた各種のデザインがあり呼び方も違う。ここでは今日の文書編集アプリケーションと電子書籍アプリケーションで周知な 2 種類の電子付箋について，その登録過程を基に特徴を確認する。

画像資料を詳細に分析する場合，モニタで表示を拡大しながら気づいた箇所があれば電子付箋に

よって記録する有用性に異論は少ないと思うが，従来このためには先ずマウスを主としたポインティングデバイスを用いてモニタに表示した画像上に次のような手順で登録を行なう。

(1) 目的の箇所を表示・拡大し，

(2) アプリケーションの専用ツールを用いてマウスで図形（四角形やライン）を描き，

(3) (必要に応じて) この図形にコメント等のテキストをキーボードから入力する。

これらは簡単な手順であるが，実際の適用となるとポインティングデバイスの操作の習熟や精緻さと共に，大量作成時は忍耐や体力も求められる。著者の経験上モニタ画面は大型の方が作業負担が少ない。なお，標準的な Web ブラウザではこのような電子付箋機能はなく，また，デジタルアーカイブの Web サイト側も電子付箋の利用を想定して画像資料を公開していない。以上の検討は主にアプリケーション［1］［2］で PC（Personal Computer）内の資料を利用する情報環境を想定している。

次に，対象が画像ではなくテキストデータの場合について，電子書籍アプリケーション［3］に備わるラインマーカ機能をタッチ操作によって使う手順を考える。

(1) 表示されているテキストデータの文字列をタッチ選択し，

(2) 選択範囲を（ツールごとの各種方法によって）確定し，

(3) その選択範囲に与える属性の決定によってラインマーカ状の電子付箋を登録する。

(4) (必要に応じて) この電子付箋にコメント等のテキストを入力する。

タブレットデバイスでは主に指で文字列の箇所を「なぞって」選択するが，これには表示されたテキストデータの文字列範囲しか選択できない仕組みが効果的に作用している。この「拘束する道筋」の性質は電子付箋の位置や操作方向を制約することになるが，同時に操作の容易さを与えている。行をまたぐ文字列や複数行にわたる文字列の場合にも合理的な指定が行える。

❸……画像文字用の電子付箋作成のために必要なこと

画像中の文字をテキストデータのように選択したり検索できるユーザーインタフェースとしては，PDF（Portable Document Format）の透明テキストが以前からよく知られており，製作用アプリケーションやネットワーク上の類似するサービスも存在する。この原理は各種のレイアウト解析や文字認識技術を用いてあらかじめ画像内の文字ごとに座標値と，変換した文字コードを記録し，表示を行なう際にテキストを透過的に画像と重ねる仕掛けで，現在の印刷された文字については概ね実用範囲にある。

しかし，歴史研究のための資料では手書き文字や経年変化した印刷物，分野や資料ごとに特有のレイアウト構造もあり，効率良くテキストに自動解析することはまだ難しい。特別なプロジェクトでもない限り，修正のために人手を併用する文字コードへの同定は相当な時間もかかり高コストとなる。

仮に優先順位を設けて検索のための文字同定の要求はひとまず留保し，画像文字位置の対応付けのみを整備するだけでも，画像文字をテキストデータのように「なぞって」選択し，ラインマーカ状の電子付箋を作成することができる。ただし，位置データを作成する場合も前述した課題は残る。

仮に半自動化できても，一文字単位で位置を対応付ける作業［4］にはやはり手作業が必要で，基盤的な文字位置情報の準備のための根本的な解決には遠い。

電子付箋は資料の内容へのリンク対象にもなるが，タッチ操作で付与できる基盤的な環境が準備できれば，デジタル化された資料のより詳細な分析や資料相互の関連性の蓄積が進む。これを前進させるためには，自動解析技術を駆使した大規模な整備と並行して，仮に暫定的なつなぎの役割であったとしても，必要な際に必要なだけ利用できる簡素な手段も必要であると考える。

❹……テキストの動線

ここで，少しデジタルデータを離れ，紙の資料に立ち返って文字のレイアウトについて考える。

最近ではキーボードに向かうばかりで文章を手書きすることも少なくなったが，どのような言語や文化による違いがあっても，縦書きであれ，横書きであれ，多少斜めになっても文章は一定の方向に記述される。文章が行末になっても，その後も続くのであれば，次の行の先頭から再び文章が始まる。印刷物やモニタ上に表示されるテキストデータもこの点は同じである。

対象とする歴史研究のための画像資料では，図 4-1 のような各種の文字レイアウト構造がある。この図では縦書きの日本語を例として，代表的な文字列の配置例を格子や線で表現している。（A）は印刷物などの整った配置の文字レイアウト構造。（B）は文書や書簡などのように行間や文字の大きさの違いや字下げもあるが，文字列は黒い太線で示すように一定方向であり折り返して続く。毛筆の書簡などでは（C）のように複雑で意味を伴わないと文字を追えない対象もあるが，（A）から（C）に至る難易度の差はあっても多くはレイアウトを推測しながら文字を追ってゆける。

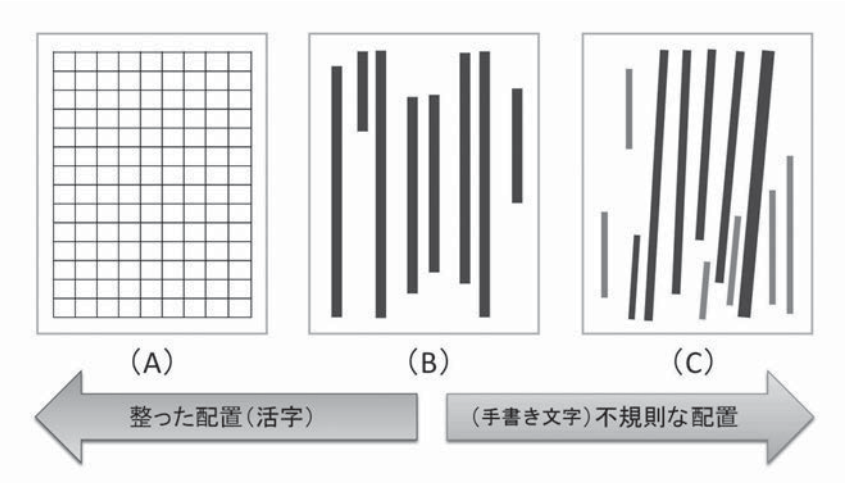

図 4-1　画像資料における文字のレイアウト（模式図）

私たちはこの「あたりまえ」なこととして習得した文字読み込みの規則に従い，たとえ各文字が読めなくても試行錯誤しながらでも，とにかく文字列を視覚的にあるいは触覚的にたどることができる。画像中の文字列をたどってゆくことは，テキストデータを選択してゆくことと同義である。

「たどる」行為には位置と時間の変化を伴うため，「動き」や「流れ」と考えることもできる。建築や都市計画には人や交通の動きを考慮した設計を行なうために動線という用語があるが，この概念を借りて，ここで検討した文字レイアウトに暗黙に存在する特徴を「テキスト動線」と名付ける。画像中の文章にはテキストデータは存在しないが，「テキスト動線」という考え方によって，画像中の文字列のレイアウトを，仮想のテキストデータに関連付けるインタフェースモデルを導くことができる。

図 4-2 はテキスト動線の概念図である。紙に書かれた文章を模している。テキスト動線は文字列に重なり，文字列の方向と長さを持つ仮想的なベクトルとして表現できる。例えば（A1, B1）の点座標の組み合わせが行単位のテキスト動線を示す。複数行を一括するテキスト動線は，［(A1, B1)，(A2, B2)］の配列構造で表せる。また，行内の文節など意味的な区切り位置や，手書き文字などで方向が変化する位置，さらには単語の区切りなども，例えば［($A1_1$, $A1_2$)，($A1_3$, $A1_4$)，($A1_5$, $A1_6$)，…，($A1_{n-1}$, $A1_n$)］のように配列として細分化して表わすことができる。ここで，$A1_n$ は B1 と同じ点座標を表わす。タッチ操作に用いるという目的上，テキスト動線に高精度な数値化は必要ではない。おおよそ文字列に重なっていれば良い。このデータモデルの拡張の前に，テキスト動線を用いた電子付箋のデザインを検討する。

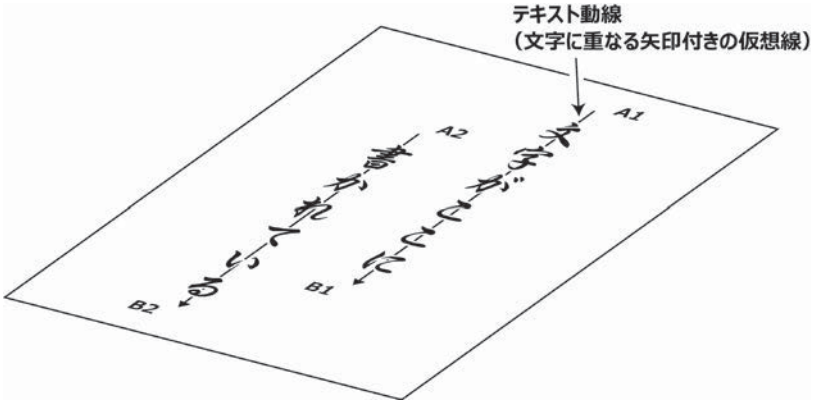

図 4-2　テキスト動線（概念図）

❺……電子付箋のデザイン

以下の説明では，Web ブラウザで HTML（HyperText Markup Language）と JavaScript を用いた表示環境を想定し用語を用いる。

テキスト動線を用いた電子付箋の作成方法について図 5-1 で説明する。簡単化のため，図 4-2 と同じく画像上に 2 行にわたり縦書きで文字が表示されていると仮定する。説明上，文字の内容は関係が無いので，図では文字自体は薄く表示してある。各行に相当する文字列の上にテキスト動線（A1, B1），（A2, B2）が仮想的に存在している。テキスト動線の各位置データは「地」である画像座標系において構造的な座標値として準備されているものとする。テキスト動線の具体的なデータ構造は後述する。

さて，マウスデバイスのポインティングやタッチ等の操作イベントを受け付けるため，テキスト動線からユーザーインタフェースの仕組みを作成する。テキスト動線の位置データを基にして，さらに適切な幅を与えた図形領域を JavaScript でオブジェクトとして作成する。このオブジェクトは本報の電子付箋の基盤となる仕組みであり，Vein（ベイン）[1]［5］と新たに用語を定義する。

この Vein（ベイン）の役割は 2 つある。

（1）疑似的なテキストデータ列として，タッチ操作等のイベントを受け付ける

（2）自身の形状を用いて電子付箋を作成する型（かた）となる

従来の電子付箋システムには無いこの仕組みを設けることで，画像の文字列を擬似的に選択することができる。Vein の図形範囲しかタッチ等の操作を受け付けないため，操作方向や範囲を拘束する道筋として，電子書籍アプリケーションのテキストデータ文字列を選択するユーザー体験と類似の効果をもたらす。Vein 自体の形状を電子付箋に用いることにも，分かりやすさと共に，電子付箋の対象や形状を事前に指定できるという副次的な利点がある。Vein（ベイン）はテキスト動線の座標データから付箋作業を行なう際に動的に作成できる。

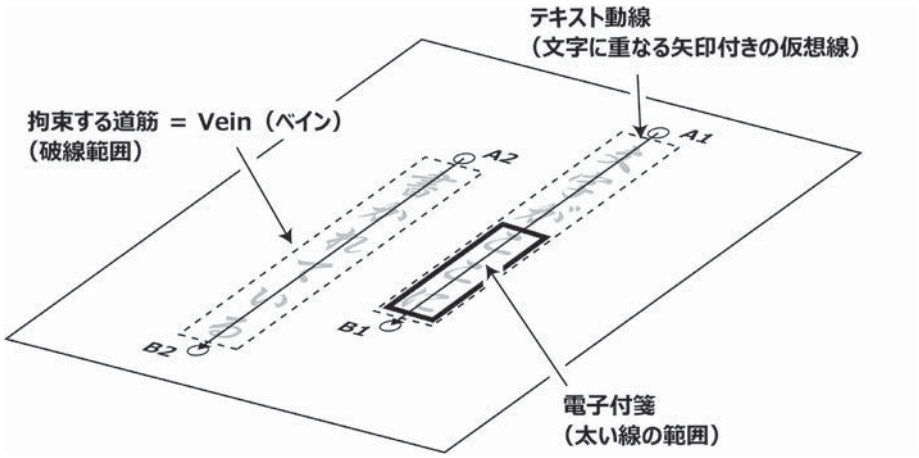

図 5-1　テキスト動線を用いた電子付箋モデル

Vein（ベイン）の配置位置はテキスト動線の座標値に依存するが，形状までは決まらない。どのような形状が付箋の作成に望ましいか検討を行なった。

図 5-2（a）は画像中の文字列に従来型の図形状の電子付箋を適用した例である。これは模式化すると（b）のように表せる。対象とする画像文字について，濃い丸印から四角印へ点線のようにマウスポインタを動かすことで対象範囲を囲む長方形の電子付箋を作成している。この電子付箋は例えば HTML では DOM（Document Object Model）要素を用いてマウス等で選択が可能なオブジェクトにできる。

このオブジェクトは検討上は電子付箋であるが，Vein として考えても同じである。このとき（c）のように文字列の方向が資料に対し斜めの場合は，その長方形の範囲は大きくなり隣接する Vein と重なって正しく選択が行えない場合が生じる。

しかし，HTML5 では SVG（Scalable Vector Graphics）を利用できるため，（d）の灰色範囲で示すように文字列の方向に沿ってポリゴン（多角形）の記述が可能である。（d）（e）のような SVG を作成するために必要なデータは，図中の実線および（e）では破線を含むポリライン（連続した線分）であり，これらはテキスト動線に相当する。

なお，電子付箋の基盤としては導入のハードルをできるだけ低くして利用できるようにしたい。そのためデータ構想はできるだけ簡潔にし，必要に応じて拡張できるような階層性を持たせることが望ましい。最低限の記述は手書きでも作成できるようにする。

テキスト動線と Vein を用いた電子付箋基盤のモデルの概要をまとめると，次のようになる。

（1）テキスト動線の座標データからタッチ選択を拘束するオブジェクト Vein を作成する
（2）Vein に対する選択操作を経て，Vein の一部あるいは全部から電子付箋を作成する
（3）テキスト動線の座標データはラインあるいはポリラインの配列構造とする
（4）テキスト動線の配列構造を基にして，連続する複数の Vein を選択可能にする
（5）可能な限り単純なデータモデルとする

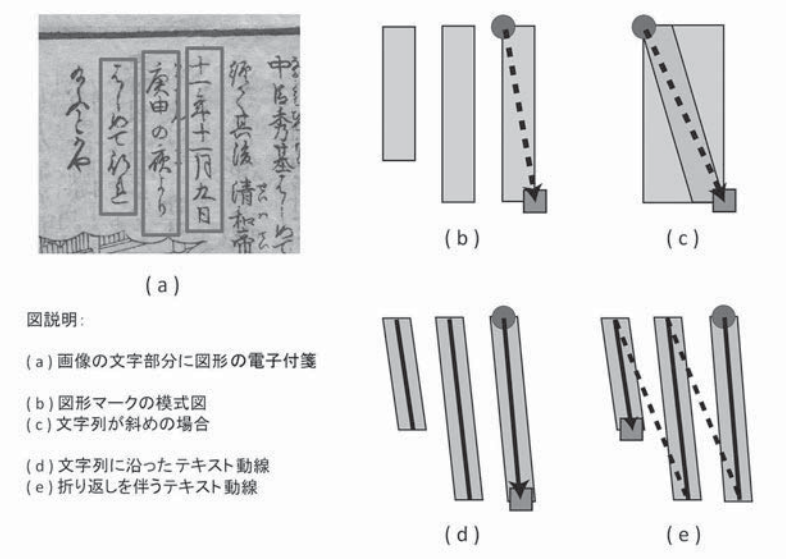

図 5-2　テキスト動線と Vein（ベイン）の形状の条件[4]

❻……テキスト動線の一般化データモデル

テキスト動線の記述のしやすさを目指して一般化データモデルを作成する。[6] 資料には多様な構造があるが，個々の行は文字列の骨格をもってポリラインで表し得る。日本語縦書き，右上から左下へ折り返しを伴う表記を仮定する。この行進行を以下 tb-rl 系とする。手書き文字の場合は図 4-1（C）のように行の向きが一様でない場合もあるが，複雑な構造であっても行を分割すると個々には単純となり，最終的に座標データの配列としてまとめ得る。

テキスト動線の検討の際に説明したように，行内の文節など意味的な区切り位置や，手書き文字な

どで方向が変化する位置，さらには単語の区切りなどに基づく行の細分化構造も含めた場合の tb-rl 系のモデルを図 6 に示す。同じく先の説明から（P1，P2），（P2，P3）などもそれぞれテキスト動線である。各座標系は Web 分野での画像の扱いにならって左上に原点，右（X）方向および下（Y）方向に増加。右上角の丸いポイントから左下の四角いポイントまでの矩形がこの場合は文字列の画像上の範囲を表わす。それらの内側のポリラインにより行や細分化した行を定義する。なお，このポリラインに沿って灰色で示されているポリゴンが Vein である。

開始点 Pa 以外はすべて開始点を基点とする相対座標とする。縦書きを想定しているため，P1 の X 値が P1 を基準とする選択の幅値の 1/2（絶対値として扱う）を与える。座標は SVG の表記にならい各ポイントの XY 座標はコンマ，各ポイントの連結は半角スペースで行い文字列として扱う。連結の順序は開始，終了点から成る矩形座標データに続けてポリラインデータを配置する。オプション的な折り返しポイントと，その他のポイントとの区別はポリライン内の Y 値の変化で判断することとした。単位は基となる画像のピクセル値とするが，比率を小数値あるいはオフセット化した整数値で用いる方法も考え得る。このような矩形とポリゴンを融合させた座標データモデルを用いることで，一般的な画像の内容には矩形部分を適用できるため，今回対象とする画像文字のみならず広く同一モデルが適用できる。

ここで，（Pa，Pb）で構成される矩形範囲に関して少し補足をしておきたい。Pa と Pb の位置関係において，常に図 6 のように範囲を表わすとは限らない。実際の範囲はポリライン部分の X 座標と Y 座標の各最大値と最小値を以て Vein 作成時に再定義する必要があり，今後の検討課題である。

テキスト動線から作成する Vein の目的は，選択範囲を拘束できる「おおまかな」道筋を与えることであって，レイアウトを精密に記述することではない。現時点のモデル化では可能な限り簡略化が望ましいと考えるため，ふりがなや割り注などの構造はこのデータモデルの積極的な記述対象としない。

データは少なくとも次のパラメータを持つ配列として構造化でき，具体的に XML や JSON 形式で記述できる。実装の章で具体例を示している。

・ID（順序の識別子）

・座標値の配列（範囲を導く矩形座標値，テキスト動線のポリライン座標値）

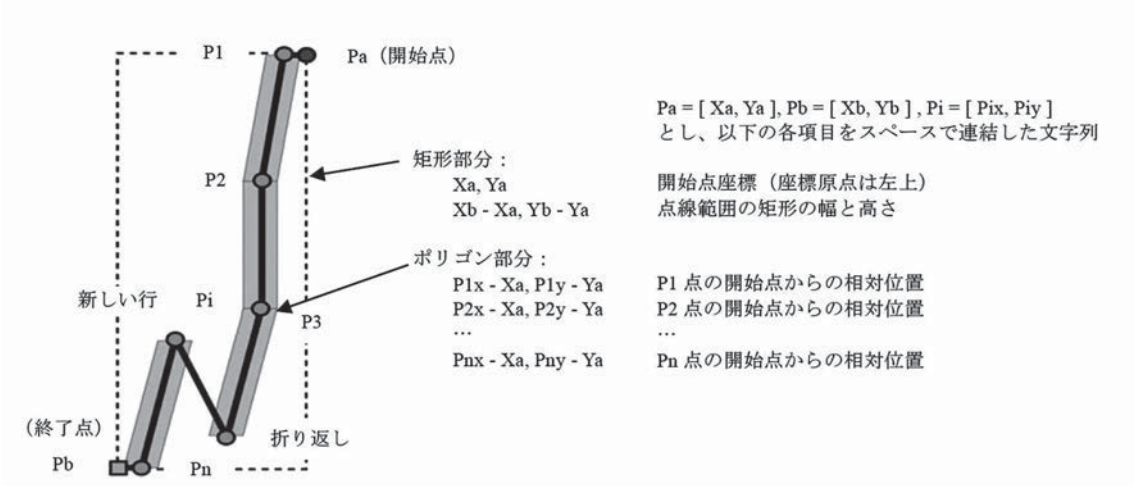

図 6　テキスト動線の一般的なデータモデル

❼……ユーザーインタフェースを考慮したデータモデルの工夫

前章で簡単に記述したが，データモデルの中にユーザーインタフェース上の工夫を 2 点織り込んでいる。ひとつは Vein を構成した際のタッチ範囲のデータである。タッチ操作を受け付ける範囲の大きさは縦書きの場合は幅が相当するが，これがモニタに表示される画像文字列の輪郭その範囲が成す幅に近いことが操作を理解する点で自然である。しかし，画像文字の輪郭範囲の大きさは特に手書き文字では値の振れも大きく，この数値の付与方法については各種考え得るだろう。自動化する場合には妥当な計算方法を考える必要もあるが，(簡単な編集ツールを用いた）手作業でのデータ製作を想定する上では，その値はおおよそ見た目に基づいても何ら問題はない。

図 7-1 は行データと共に文字の大きさを一筆書きで定義している。図 7-1（1）の縦書きの場合，そのテキスト動線の範疇にある画像文字のおおよその横幅 1/2 を最初のポリラインストロークで定義する。この値の 2 倍をタッチ操作の幅として Vein の作成データに用いる。点線の正方形を仮想的な画像文字の大きさとする。行の先頭で決めるために行の途中や折り返しのある場合は文字の大きさが変わることもあるが，行の終端まで同じ値を用いる。手作業時は極端なものは予め目分量で程度を考慮すれば良い。また，隣接する行で大きく変わる場合は，別のテキスト動線として作成すれば良い。いずれにしてもタッチ操作の幅は隣り合う行で Vein の重なりが生じないように作成する必要がある。

図 7-1（2）の欧文横書きのレイアウトではタッチ操作の高さを与える必要があるが，縦書きの場合と類似構造でテキスト動線は定義できる。各種言語の手書き文字に対応できると予想する。

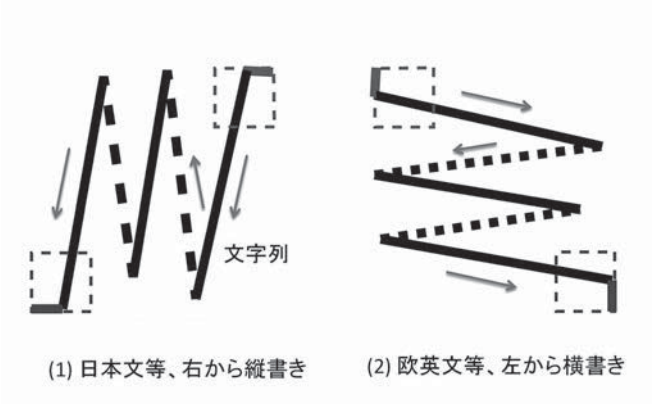

図 7-1　文字列方向への対応と文字サイズ

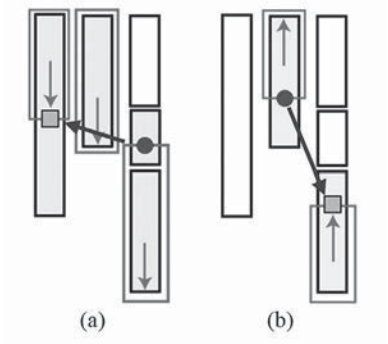

図 7-2　連続選択

もうひとつの工夫として，テキスト動線を利用した行や文節などによる行内の区切りをテキストデータのように連続的に選択する方法について説明する。図 7-2 で，各 Vein を濃い黒色の枠とする。これら SVG のオブジェクトはプログラム内では順序を持つ配列である。選択する際，タッチした開始点（丸印）からリリースする終了点（四角印）まで指でなぞる（濃い矢印）と，両端のポイント座標と SVG オブジェクトの順序番号が得られる。この順序の増減に従って簡単なアルゴリズム

で範囲にある SVG 要素の表示スタイル（例えば背景色）を動的に更新する。外側の薄い枠線の箇所がこうして得られた選択結果であり，登録すれば電子付箋となる。リリース時の順序値がタッチ時の順序値より大きければ図 7-2（a）のように後方への選択，逆に小さければ図 7-2（b）のように前方と解釈する。この場合も，欧文横書きにおいて類似に考えることができる。

❽……HTML5 による実装

以上のデザイン検討を基に，タブレットデバイスで電子付箋のインタフェースを実現する方法について説明する。デジタルアーカイブ画像が Web 上で多く公開されている現実を考えると，各 OS（Operating System）のネイティブアプリケーションよりもデバイスのデフォルトのブラウザで Vein（ベイン）のユーザーインタフェースを実現できることが望ましいと考えた。その場合，HTML5［7］と CSS3［8］そして JavaScript が使える道具である。この環境[(2)]と次のような工夫で簡単なプロトタイプを作成し可能性検証を行った。［5］

（1）HTML の DOM を CSS で絶対配置して，Vein 用，電子付箋の編集用，登録付箋の保持用の 3 種類のレイヤ構造を作る
（2）テキスト動線の Vein 実装は SVG で行う
（3）JavaScript で CSS を制御し，各レイヤの表示状態を切替える
（4）登録した電子付箋のデータ保存は，プロトタイプではサーバーに依存しないよう HTML5 の機能であるローカルストレージ（Local storage）を用いる

HTML5 を用いることで，従来の PC を含め各種の情報デバイスで利用できる。以下では実装手順や操作による変化を図を用いながら順を追って説明する。

図 8-1 にテキスト動線，図 8-2 にそのデータ例を示す。画像資料の上にテキスト動線のデータを描いた別のレイヤをかぶせて表示した様子である。例として右から行単位，複数行の一筆書き様式，文節位置における行内細分化構造の 3 種類のテキスト動線の画像文字への重なり方を示している。なお，テキスト動線は Vein を作成するための仮想線（ベクトル）で実際には表示することはない。

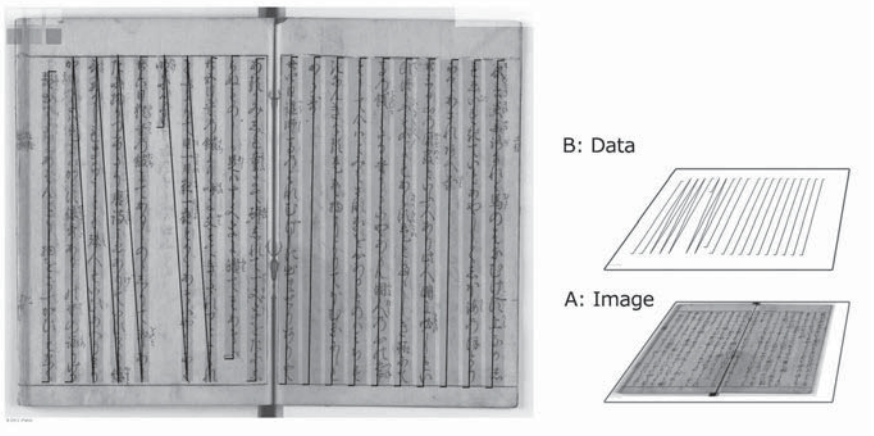

図 8-1　テキスト動線の仮想線（実線部分）[(4)]

```
[
① {"type":"text","data":"1804,208  -68,1179  -23,  0  -44,1179","id":"T1"},
  {"type":"text","data":"1721,206  -67,1180  -23,  0  -44,1180","id":"T2"},
  {"type":"text","data":"1639,205  -68,1179  -24,  0  -45,1179","id":"T3"},
  {"type":"text","data":"1556,203  -68,1180  -23,  0  -44,1180","id":"T4"},
  {"type":"text","data":"1473,202  -67,1179  -23,  0  -44,1179","id":"T5"},
  {"type":"text","data":"1391,201  -68,1179  -24,  0  -45,1179","id":"T6"},
  {"type":"text","data":"1308,199  -68,1180  -24,  0  -45,1180","id":"T7"},
  {"type":"text","data":"1225,198  -68,1179  -23,  0  -44,1179","id":"T8"},
  {"type":"text","data":"1142,196  -67,1180  -23,  0  -44,1180","id":"T9"},
  {"type":"text","data":"1060,195  -68,1179  -24,  0  -45,1179","id":"T10"},

① {"type":"text","data":"938,186  -63,1162  -28,  0  -31,1162","id":"T11"},
① {"type":"text","data":"847,177  -53,1091  -26, -1  -22,1091","id":"T12"},

② {"type":"text","data":"767,185 -218,  249  -20, -4  -25,1167
                         -107,-10 -108,1164
                         -190,-12 -193,  249","id":"T13"},

③ {"type":"text","data":"516,179 -299,1173  -25, -1  -23,  157  -20,1174
                         -107, -7 -110,  476 -105,1111 -105,1177
                         -184,  0 -195,1176
                         -269, -1 -274,  526 -272,1173",          "id":"T14"},

① {"type":"text","data":"187,233  -53,1119  -28, -1  -28,1119","id":"T15"}
]
```


図 8-2　テキスト動線データ例（JSON）

図 8-3 はテキスト動線データから作成した Vein（着色部分）を HTML5 の SVG で実装した説明図である。テキスト動線が斜めであっても選択のための一定の幅を保ち，各 Vein の重なり無く選択できる。B のテキスト動線のレイヤは仮想的で実際には表示も利用もしない。C のレイヤは図では各行に重なる濃い筋であるが，これが Vein である。このレイヤは通常は透明化しておくことで，操作をしなければ単に画像のみ表示の状態となる。そしてこの状態が，本稿の電子付箋利用の基盤となる。

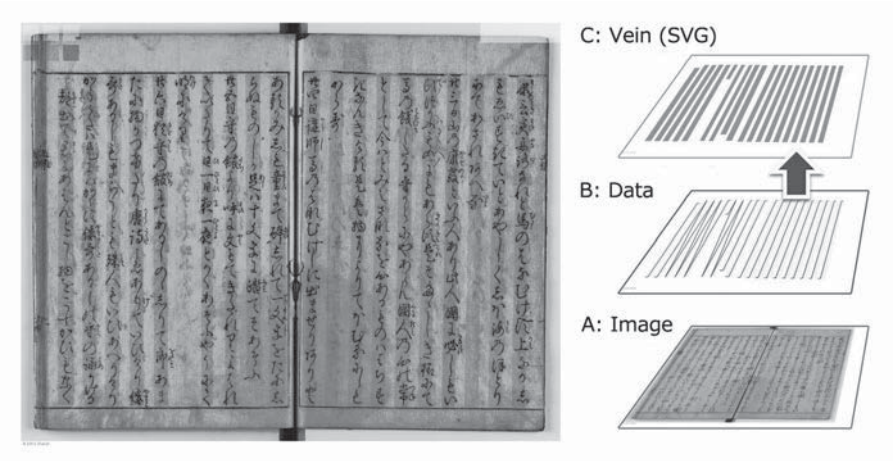

図 8-3　テキスト動線の SVG による実装（文字に重なった着色部）[(4)]

図 8-4 では Vein を最上部の C:Vein レイヤに配置し透明化した上でタッチ操作関連のイベントを設定する。イベント結果を直下の D:Work レイヤで E:Mark として表示する。この E:Mark は電子付箋に相当し Vein からデータを受け継いだ SVG であるが，この段階では表示の目的のみとする。図ではこの電子付箋を登録するかキャンセルするかを選ぶだけの簡単な登録確認用のツールボックスを表示している。必要に応じメタデータ等のテキストを入力する仕組みを実装する。図 8-6 にテス

ト的に実装したローカルストレージ上の登録結果を示す。

なお，付録のアプリケーションではタッチの選択を何度でもやり直すことができるように，タッチして一定時間経過後に電子付箋の登録確認用ツールボックスを表示する。

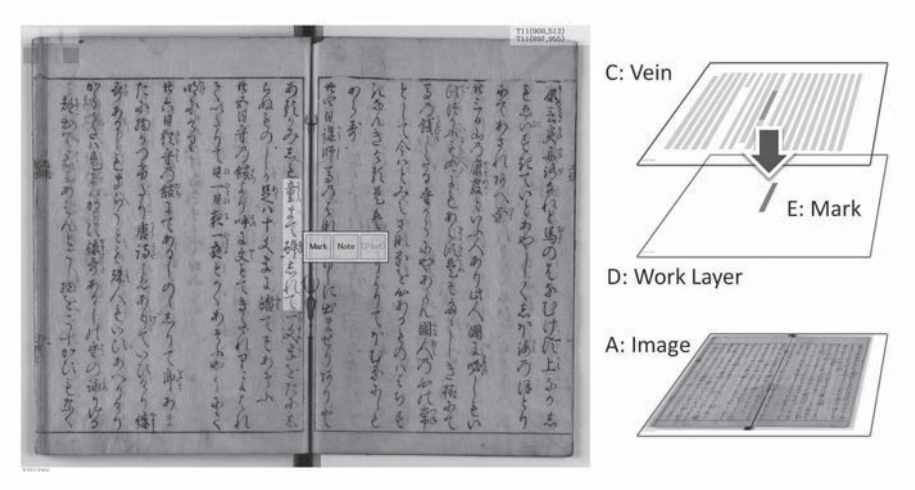

図 8-4　Vein を用いた選択時のインタフェース構造(4)

最後に，図 8-5 は電子付箋の利用時に登録データを表示する場合である。C:Vein の情報を電子付箋形状の型として使い，登録した座標データから F:Note（App.）レイヤに再現する。この再現した電子付箋の各要素にタッチ関連イベントを設け，再編集などの機能を実装できる。C:Vein レイヤと D:Work レイヤは非表示とすることでイベントの伝搬を制御し，結果として登録済の電子付箋に対しタッチ操作が行える。

付録の実装例では，この利用時と先の登録時の状態は切替えボタン等を設けて操作上の区別を行なっている。

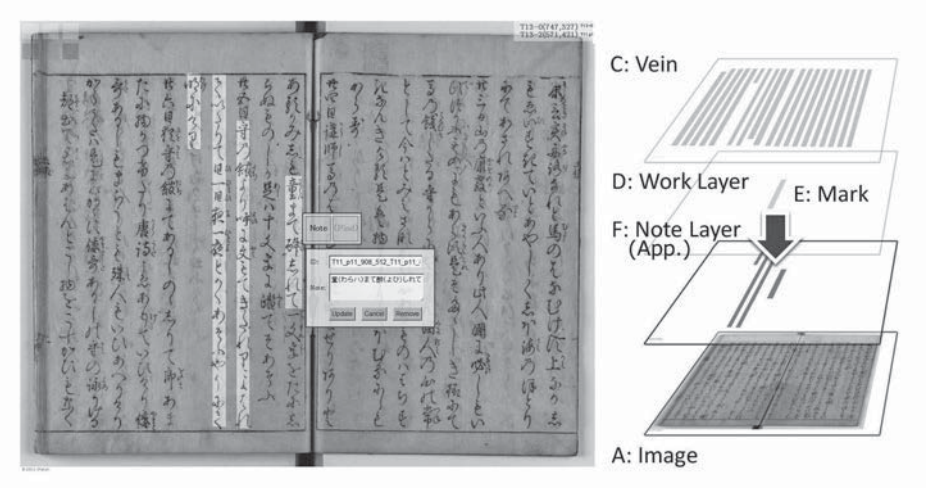

図 8-5　登録済電子付箋の表示(4)

```
"tosa03_T11_p11_908_512_T11_p11_897_955":
{
"p1":"T11_p11","p1x":908,"p1y":512,
"p2":"T11_p11","p2x":897,"p2y":955,
"maxrect":[880,512,57,443],
"areaval":25251,
"type":"spot",
"url":"http://localhost:8080/vein/pages/tosa03.html",
"image":"../images/tosa03.jpg",
"vinedata":"tosa03.json",
"description":"童(わらハ)まて醉(よひ)しれて",
"createdate":"2012-10-08",
"public":"true",
“identifier":"T11_p11_908_512_T11_p11_897_955"
}
```


図 8-6　ローカルストレージ（Local storage）に記録するデータ例

❾……テキスト動線データの作成

評価・検証用のテキスト動線データは，著者の既存の電子付箋編集アプリケーション［2］を拡張して作成した。作成方法は一般の図形アノテーション編集方法と同様であり，従来型の電子付箋をポリラインで作成してからテキスト動線のデータの形式に変換した。規則的な文字配列のためにグリッド型の一括編集ツールも試験搭載した。データモデルは矩形とポリラインの複合構造のため，挿絵，頭注などは矩形情報を用いて選択箇所を簡素化できる。編集過程で座標の修正や行の連続選択のための順序を変更できる。このアプリケーションの情報に関しては付記に記載する。

テキスト動線のデータ記述仕様は簡易化を目指したが，対象とする画像のレイアウトが縦方向や横方向に直交する場合には，Vein を矩形データ部分のみから作成できる。このような対象に限った編集アプリケーションを作成することは難しくはなく，Web アプリケーションに実装することも可能である。なお，簡易化には別の意図もある。データ記述が簡単であれば，各種の公開された画像レイアウト解析の技術を適用してデータを作成しやすいと考えるためである。このような編集用のライブラリを待望する。

❿……操作評価

今回の試作実装では，主にタブレットデバイス［9］におけるタッチ操作を対象にしたが，同時に従来の PC とマウスデバイスによる操作にも対応した。試作は実際には，データモデル化，実装，評価を独立して行なったわけではなく，同時に相互に修正を加えながら進めた。従って，操作上の課題は先送りか代替策で補っている。ただし，タブレットデバイスを対象とする上で今後の検討課題もある。それらを 3 つ取り上げる。

10.1. 操作範囲の大きさの影響

先ず，タブレットデバイスの大きさと電子付箋作成の関連性について考察する。2012 年の試作当時[(3)]，入手し得るタブレットデバイスは大きくても 10 インチ程度で，画面の面積比，画素数共に普及的なノート型 PC の 1/2 以下の情報しか表示できない。Web ブラウザを用いるために表示範囲はさらに狭くなる。従ってタブレットデバイスではタッチ操作箇所の大きさが小さくなり，操作する際に対象箇所を大きく拡大する必要が生じた。電子付箋を作成する場合には，背景の画像がタッチ操作イベントを検知しない設定にし電子付箋だけの検知を行えるようにする。横置きしたタブレットデバイスで縦書きの画像文字の場合，対象を拡大して上下方向に長い文字列を選択したい場合は終点あるいは始点が隠れ困難となる。タブレットの場合は，デバイスの方向を回転することができ，対象に応じて操作範囲を大きくする方法もあるが，実用上は電子付箋の編集時でも複数タッチの検出を伴う画像移動の仕組みや，表示端部での自動スクロール，電子付箋の範囲の再編集機能などが必要になると思われた。一方で，図 7-2 の隣接する複数の行を選択する方法は，マウス，タッチ選択ともに良好で，特に行の前後方向のどちらへでも選択できる実装は気持ちよく操作ができた。これを応用して複数ページをまたぐ選択なども実装が行えると考える。

タブレットデバイスは小さな画面であるからこそ，PC に比べて利用の範囲が拡がるのであり，むしろ小さな表示範囲でどのようにタッチ操作や各種のジェスチャーを組み合わせて電子付箋利用の環境を提供できるかが重要である。

10.2. タッチ操作による Vein 上の位置決め精度

タッチ操作ではマウスのようにポインタアイコンで指し示すことが原理的に行えない。これに伴うマウスオーバーなどの操作イベントも存在しない。従って，タッチ操作の際にはその対象は指に隠れて見えなくなる。タブレットデバイスだけでなく，その他のタッチ操作デバイス，特に小型のデバイスには共通かつ重要な課題である。特に指でタッチする位置の精度に対しては，一般には指の下に隠された部分をタッチ位置の上側に表示する対策が行なわれるが，今回の試作ではそのような補助表示は実装していない。

電子付箋を定義する際に Vein 上で始点と終点を得るが，縦書き用では X 方向は拘束のため無視できる。両点は Vein の順序 ID と一次元（この場合 Y 方向）の位置情報だけで決まる。テキストデータの場合には各文字（キャラクタ）単位で選択の位置が決まるが，Vein のタッチ操作では画像文字列の方向にはテキストデータのような区切り用のデータを定義していないので，連続する位置の曖昧さを除くことはできない。だが，この問題に関しては曖昧であっても良いと考えている。おおよその位置に電子付箋が作成できれば目的としては充分であろう。むしろ，文字ごとの区切りが存在すると，指位置の補助的な表示が必要になる。より細かな操作の場合にはタッチペンを用いるなどの選択肢もある。簡素化を目指すデザインは複雑になるばかりの操作インタフェースに歯止めをかける意味でも今後大切なテーマであるかもしれない。

10.3. 電子付箋の重なり

一般的に電子付箋では作成した図形が重なる課題があり，登録した電子付箋の利用時に影響が現れる場合がある。本稿で提案した電子付箋では Vein によって作成範囲が拘束されるため，従来の図形状の電子付箋をマウスポインタで選択する場合よりも選択が難しい。しかも先に考察したタッチ位置の曖昧さの課題も関係する。

この課題に対しては，電子付箋の面積情報（画素基準）を用いてソートを行ない，最も面積の小さな電子付箋オブジェクトから先にタッチ位置の判定を行なうように考えた。

なお，関連事項としては，作成した電子付箋は CSS で透過的な着色を行ない下層レイヤの画像文字を表示しつつ電子付箋の判別の補助としているが，電子付箋自体が重なりと共に表示箇所が濃くなるが，同時に文字が見え難くなる。画像の乗算的な機能を HTML5 の Canvas API（Application Programming Interface）を使って実現する方法も将来的には考慮する余地がある。

⓫……まとめ

デジタルアーカイブ化された画像資料，その中で文字が主体の資料に対し，タブレットデバイスでの利用を主に想定して電子付箋の利用を促進するための仕組みの提案を行なった。先ず，テキスト動線という画像上の文字列とテキストデータをつなぐ概念を構想した。次にテキスト動線に基づき，Vein（ベイン）という電子付箋の型であり，作成インタフェースも兼ねるオブジェクトをデザインした。この Vein は擬似的･仮想的なテキストデータ列であるが，これを画像文字に重ねることで，画像でありながら文字列箇所を電子書籍アプリケーションのラインマーカと同じように指で「たどって」選択できる仕組みを実現できた。テキスト動線や Vein のデータモデルを設計し，HTML5 とその周辺技術によって新しい電子付箋の基盤を実装し，試作プログラムで操作評価を行なった。

プログラム試作とその検証においては，特定のタブレットデバイスと Web ブラウザを用いた。HTML5 の実装が Web ブラウザごとに異なるうえ，タブレットデバイスの操作も特にマルチタッチで違いがある。2012 年に主な機能実装を行なったが，その時点ではまだ不十分な反応も多かったため，試作プログラムはできるだけ共通な機能で行う必要があった。

最後に繰り返しになるが，国内でデジタルアーカイブ化された多くの画像資料については，和洋を問わず古文書などから古い手書きの資料，もっと近代の印刷された書物に至るまで，画像上の文字とテキストデータの位置が対応しておらず，テキスト主体の電子書籍のようなユーザーインタフェースの体験を利用者に提供できる状態にない。そこで，このような考え方もあることを紹介する目的で，未整備かつ一部であるがテキスト動線のデータ部と Vein を中心にソースコードを資料として添付する。本稿の評価を行なった簡単なサンプルプログラムも付記に URL を記載するので，興味があればご批判いただければと思う。そう遅くないうちにプログラム共有サイトなどでも公開できればと考えている。

謝辞

共同研究者の皆様，特に安達文夫教授，鈴木卓治准教授には，テキスト動線の構想や名称定義にあたりご意見をいただきました。本アイデアや試作物に関し，個人的に貴重なご意見やご指摘をいただいた皆様へと共にこの場を借り感謝申しあげます。

註

（1）――Vein（ベイン）とは体の静脈の意味で，葉脈，鉱脈などの訳も有する。画像に暗黙に在るテキスト動線インタフェースとの相似を感じこの名称を選んだ。

（2）―― 本稿のプログラムの再試作時点の HTML5 や CSS3 は参考文献［7］［8］が該当するが，実際の検証用には主として当時の Google Chrome ブラウザを用いている。HTML5 や CSS3 の仕様とブラウザでの実装には違いがある。あくまで参考程度に記す。

（3）――2013 年に本稿作成のために，再試作を行なっている。

（4）――本稿掲載の説明図に用いている資料は，図 5 は著者個人蔵，図 8 は個人蔵である。

参考文献

［1］Adobe Acrobat X Pro, http://www.adobe.com/products/acrobatpro/.
［2］iPalletnexus, http://www.ipalletnexus.org/.
［3］Amazon Kindle, http://ja.wikipedia.org/wiki/Amazon_Kindle.
［4］津田光弘 : 構造化アノテーションによる文字画像への活字配置とその応用 , 人文科学とコンピュータシンポジウム , pp.195–198, 2004.
［5］津田光弘 :「画像文字選択のための「テキスト動線」編集・利用システム―Sijima」, 人文科学とコンピュータシンポジウム論文集 , pp.185–190,（2012）.
［6］津田光弘 : 無名付箋の画像資料への適用方法について , 情報処理学会研究報告 , Vol.2012–CH–95 No.1, 2012.
［7］W3C Candidate Recommendation 6 August 2013,　http://www.w3.org/TR/2013/CR–html5–20130806/.
［8］W3C Working Draft 20 June 2013,http://www.w3.org/TR/2013/WD–css–masking–20130620/.
［9］Apple Inc., iPad（4th Generation）& iPad mini, with iOS6.

（イパレット，国立歴史民俗博物館共同研究員）

（2014 年 1 月 7 日受付，2014 年 7 月 28 日審査終了）

A Design and Implementation of the Digital Annotation Basis on an Image Resource for a Touch Operation

TSUDA Mitsuhiro

This paper describes a user interface design and implementation to make the position of letters and line series in digital images of historical documents selectable like text lines of an electronic book. The aim is to prepare a basis by which a tablet device can be used to make a large quantity of electronic annotations.

At first, a "flow" concept, "Primitive Lead," was found as a basic and tacit property of letter images by analyzing electronic annotations, marker tools for electronic book applications and the layout of letters in image resources. Primitive Lead links the layout of letters in an image with text data.

Then, from the Primitive Lead concept, an electronic annotation basis for pseudo and virtual text data lines, "Vein," was designed. This annotation basis has the following characteristics: (1) It provides a restricted touch-and-select user interface; (2) It becomes a template by which to generate annotations itself; (3) It can support the writing layouts of each language, such as top to bottom or left to right; (4) It is compatible with various line layouts using SVG in HTML5; (5) It allows continuous selection of multiple lines; (6) It is a simple data structure for creating data by hand.

Finally, from the design of the general purpose data model, a prototype program of the above design model was implemented using HTML5, CSS3 and JavaScript. The usefulness of the prototype's user interface was then verified using a web browser on a touch-screen tablet device and on a conventional mouse-operated personal computer, and any user interface problems were investigated. The JavaScript core-application code and the URL of a guide to the Primitive Lead editing tool, with sample, are included in the appendix.

Key words: Image, Annotation, Touch Device, User Interface

付録

A. コアプログラム（JavaScript）

```
// Vein Concept Application
// @ ライセンス : 作成者はこのバージョンのプログラムのコード（以下，このコード）について著作権の行使を控える。
// 作成者は第三者によるこのコードおよび派生物の適用ならびに実行等により生じた結果や損害についてその責任を一切負わない。
// @ バージョン : 0.7（beta）
// @ 公開日 : 2013-11-07
// @ 作成者 : 津田 光弘
// @ 付記 : このコードでは基本的な Vein 概念の実装例を目指し，
// プログラム言語は公開日当時知り得る JavaScript，特定バージョンの jQuery ライブラリを想定しているが，
// 機能や変数名についてはこれらの条件や範囲に拘束されるものではない。
// UI（User Interface）については最低限の参考コードであり，実利用では更なる工夫を要する。

(function(window, $){

  var img_wid = 0;    // 画像 ( ソース ) の幅
  var img_hei = 0;    // 画像 ( ソース ) の高さ
  var vscale = 1.0;    // 画像に対する画面（Vein ビューポート）の大きさの比率（規定値 =1，等倍）

  var sk_layer = null;  // テキスト Vein レイヤ ( スケルトン ): SVG による TextVein 確認レイヤ：参考
  var ap_layer = null;  // アプリケーションレイヤ : 付箋 (Vein に基づく Spot 付箋 ) の表示レイヤ
  var wk_layer = null;  // 編集レイヤ : 付箋の作成用
  var vein_layer = null;  // Vein レイヤ : Svg による Vein 実装レイヤ

  var vein_list = null;  // JSON 形式 Vein の解析済み Vein データ配列
  var wk_cv = null;    // ターゲット Vein オブジェクト（解析用，現在の Vein）
  var ap_cv = null;    // アプリケーションレイヤーのターゲット

  var apps = {};      // 登録済みの Vein オブジェクト（Spot 付箋）

  var Util = {};      // 内部用ユーティリティ関数

  // 外部用関数
  var Vein = {
    init: null,      // 起動関数
    updateScale: function(t){    // サイズ更新（参考オプション）
      vscale = t;
    },
    inuse: false,    // Vein レイヤ利用時
    inapp: false,    // アプリケーション利用時
    waketime: 2000,    // タッチ操作時，登録ツールの起動閾時間，ms
    veinobj: {},    // 編集用オブジェクト
    UI: {
      enrty: null,    // 編集用関数（登録）
      cancel: null,    // 編集用関数（キャンセル）
      remove: null,    // 編集用関数（登録削除）

      deleteMem: null,  // 編集用関数（ローカルストレージ削除：参考）
      deleteMemAll: null,  // 編集用関数（ローカルストレージ全削除：参考）

      useApp: null,    // メニュー関数（Vein レイヤの利用，付箋利用と再編集：参考）
      useVein: null,    // メニュー関数（Vein レイヤの利用，付箋作成：参考）
      hideVein: null    // メニュー関数（Vein レイヤを非表示：参考）
    }
  };

  if (typeof window.Vein === "undefined") window.Vein = Vein;

  // 関数
  var ch_ck = function(){
```

```
  if (arguments.length == 1){
    var d = arguments[0];
    return (d != null && typeof d !== "undefined");
  }
  else if (arguments.length == 2){
    var d = arguments[0];
    var k = arguments[1];
    return (d != null && typeof d !== "undefined") && (d[k] != null && d[k] !== "undefined");
  }
  else{
    return false;
  }
};
// タッチイベント補正
//var _clientX = function(event) { return 'touches' in event.originalEvent ? event.originalEvent.touches[0].clientX : event.clientX; };
//var _clientY = function(event) { return 'touches' in event.originalEvent ? event.originalEvent.touches[0].clientY : event.clientY; };
var _pageX = function(event) { return 'touches' in event.originalEvent ? event.originalEvent.touches[0].pageX : event.pageX; };
var _pageY = function(event) { return 'touches' in event.originalEvent ? event.originalEvent.touches[0].pageY : event.pageY; };
var _elemX = function(event, e_offset) { return _pageX(event) - e_offset; };
var _elemY = function(event, e_offset) { return _pageY(event) - e_offset; };

// 補助関数
var removePx = function(s) {
  if (s == "" || s == "0") { return 0; }
  if (s.indexOf("px") > 0) { return parseFloat(s.substring(0, s.length - 2)); } else { return 0; }
};
var addPx = function(v){ if (v == "") { return "0"; } else { return v + "px"; } };

var Util = {
  getVeinList: function (o_list) {
    var vein_list = [];    // Vein 情報の配列

    var o_svg = {"id": "svg","type": "svg","width": img_wid, "height": img_hei};
    vein_list.push(o_svg);

    var ser = 0;      // Vein 配列内の順序数
    for (var i = 0; i < o_list.length; i++){
      var o_line = o_list[i];
      if (ch_ck(o_line, "type")){
        // 例：
        // {"type": "text","data": "1807, 190 -67, 1180 -23, 0 -44, 1180","id": "T1"}
        // "data" 内区切りは半角スペースが優先で点の区分，半角コンマが xy 座標値の区分。
        // このバージョンの場合，座標値は画像の左上を原点 (0, 0) とする整数系の画素値としている。
        // 比率値への拡張やプロパティーの設定は要実装のこと。
        var d_id = o_line["id"];
        var d_type = o_line["type"];
        var d_data = o_line["data"];
        // 実用には要存在チェック
        if (d_type === "vector" || d_type === "rect"){
          // 1 行パタン
          var data1 = d_data.split(/[\s,]/);
          // RL-TB（和書，右から左，上から下）
          // Rect（範囲の原点 (xr, yr)，幅と高さ (wr, hr)）
          var xr = parseInt(data1[0]) + parseInt(data1[2]);
          var yr = parseInt(data1[1]);
          var wr = -parseInt(data1[2]);
          var hr = parseInt(data1[3]);
          // Vector（行の原点 (x, y)，幅と高さ (w, h)）
          var x = parseInt(data1[0]) + parseInt(data1[4]);
          var y = parseInt(data1[1]) + parseInt(data1[5]);
          var w = parseInt(data1[6]) - parseInt(data1[4]);
          var h = parseInt(data1[7]) - parseInt(data1[5]);
          // 幅オフセット
```

```
    var x_offset = -parseInt(data1[4]);
    var y_offset = parseInt(data1[5]);
    if (x_offset > y_offset * 4) y_offset = 0;

    var tex = "";  // 割り当てテキスト
    var cnum = 0;  // 割り当てテキストの文字数
    if (ch_ck(o_line["text"])){
      tex = o_line["text"];
      cnum = tex.length;
    }
    var lnum = 1;  // 行数
    var url = "";  // リンク
    ser = ser + 1;  // Vein 配列内の順序数

    var obj = {
      "ser": "p" + ser,
      "id": d_id,       // 文字列
      "type": "polygon",
      "di": "TB",       // RL-TB（和書，右から左，上から下）の略
      "rect": [xr, yr, wr, hr],
      "vector": [x, y, w, h],
      "offset": [x_offset, y_offset],
      "points": d_data,  // RESERVED
      "ex": "",       // RESERVED
      "text": tex,    // RESERVED
      "line": lnum,
      "url": url       // RESERVED
    };
    vein_list.push(obj);
  }
  else if (d_type === "text" ¦¦ d_type === "polyline") {
    // 複数行パタン
    var cnums = 0;
    var data1 = d_data.split(/[\s,]/);
    // RL-TB
    // ボックス原点
    var xb = parseInt(data1[0]);
    var yb = parseInt(data1[1]);
    // 幅オフセット
    var x_offset = -parseInt(data1[4]);
    var y_offset = parseInt(data1[5]);
    if (x_offset > y_offset * 4) y_offset = 0;

    if (data1.length > 8) {
      // 複数行パタンと判定
      var tex = "";
      var texs = [];
      if (ch_ck(o_line["text"])){
        tex = o_line["text"];
        texs = tex.split(/[\r\s]/);
      }
      var k = 0;
      var hmax = 0;
      var lnum = 1;
      for (var j = 4; j < data1.length - 3; j = j + 2){
        if (parseInt(data1[j + 3]) > parseInt(data1[j + 1])) {
          // Vector
          var x = xb + parseInt(data1[j]);
          var y = yb + parseInt(data1[j + 1]);
          var w = parseInt(data1[j + 2]) - parseInt(data1[j]);
          var h = parseInt(data1[j + 3]) - parseInt(data1[j + 1]);
          // Rect
          if (w < 0){
```

```
        var xr = x + w - x_offset;
        var yr = y - y_offset;
        var wr = -w + 2 * x_offset;
        var hr = h + 2 * y_offset;
      } else {
        var xr = x - x_offset;
        var yr = y - y_offset;
        var wr = w + 2 * x_offset;
        var hr = h + 2 * y_offset;
      }

      var tex = texs[k];
      var cnum = 1; //tex.length;
      var url = "";
      ser = ser + 1;
      var obj = {
        "ser": "p" + ser,
        "id": d_id + "-" + k,
        "type": "polygon",
        "di": "TB",
        "rect": [xr, yr, wr, hr],
        "vector": [x, y, w, h],
        "offset": [x_offset, y_offset],
        "points": d_data,
        "ex": "",
        "text": tex,
        "line": lnum,
        "url": url
      };
      vein_list.push(obj);

      cnums = cnums + cnum;
      if (hmax < y + h) hmax = y + h;

      k = k + 1;
    } else {
      lnum = lnum + 1;
    }
  }
}
else{
  // 1 行パタン（type: rect と同様）
  // RL-TB
  // Rect
  var xr = parseInt(data1[0]) + parseInt(data1[2]);
  var yr = parseInt(data1[1]);
  var wr = -parseInt(data1[2]);
  var hr = parseInt(data1[3]);
  // Vector
  var x = parseInt(data1[0]) + parseInt(data1[4]);
  var y = parseInt(data1[1]) + parseInt(data1[5]);
  var w = parseInt(data1[6]) - parseInt(data1[4]);
  var h = parseInt(data1[7]) - parseInt(data1[5]);
  // 幅オフセット
  var x_offset = -parseInt(data1[4]);
  var y_offset = parseInt(data1[5]);
  if (x_offset > y_offset * 4) y_offset = 0;

  var txr = xr;
  var twr = wr;
  if (w > 0){
    txr = x - x_offset;
    twr = w + 2 * x_offset;
```

```
            }

            var tex = "";
            var cnum = 0;
            if (ch_ck(o_line["text"])){
              tex = o_line["text"];
              cnum = tex.length;
            }
            var lnum = 1;
            var url = "";
            ser = ser + 1;
            var obj = {
              "ser": "p" + ser,
              "id": d_id,
              "type": "polygon",
              "di": "TB",
              "rect": [txr, yr, twr, hr],
              "vector": [x, y, w, h],
              "offset": [x_offset, y_offset],
              "points": d_data,
              "ex": {},
              "text": tex,
              "line": lnum,
              "url": url
            };
            vein_list.push(obj);
          }
        }
      }
    }
    return vein_list;
  },
  checkInOut: function(_list, _x, _y){
    // 座標（_x, _y）の内部存在判別
    // 一次選別
    var target = [];
    for (var i = 1; i < _list.length; i++){
      var rect = _list[i]["rect"];
      target.push(_list[i]);
    }
    // 二次選別
    if (target.length < 1){
      // 無し
      wk_cv = null;
    } else {
      for (var i = 0; i < target.length; i++){
        var d_rect = target[i]["rect"];
        var d_vector = target[i]["vector"];
        var d_offset = target[i]["offset"];

        if (d_rect.length === 4 && d_vector.length > 0){
          // 内部判定（要再確認）
          var b = false;
          var p = getSVGPointsArray(d_rect, d_vector, d_offset, null);

          var c1 = (p[1][1] - p[0][1]) * (_x - p[0][0]) <= (p[1][0] - p[0][0]) * (_y - p[0][1]);
          var c2 = (p[2][1] - p[1][1]) * (_x - p[1][0]) <= (p[2][0] - p[1][0]) * (_y - p[1][1]);
          var c3 = (p[3][1] - p[2][1]) * (_x - p[2][0]) <= (p[3][0] - p[2][0]) * (_y - p[2][1]);
          var c4 = (p[0][1] - p[3][1]) * (_x - p[3][0]) <= (p[0][0] - p[3][0]) * (_y - p[3][1]);

          b = !(c1 && !c3) && !(c2 && !c3) && !(c2 && !c4);

          if (b) {
```

```
            wk_cv = target[i];
            break;
          }
        }
      }
    }
  }
};

// Vein の svg 作成用コンストラクタ
function Svg(){
  this.NS = "http://www.w3.org/2000/svg";
  //this.draggin = false;
};
Svg.prototype.createSvg = function(jq_obj, data_list){
  if (ch_ck(data_list)){
    var svg = document.createElementNS(this.NS, data_list[0]["type"]);
    svg.setAttribute("width","100%");
    svg.setAttribute("height","100%");
    svg.setAttribute("viewBox","0 0 " + img_wid + " " + img_hei);
    for (var i = 1; i < data_list.length; i++){
      svg.appendChild(this.getSvg(i, data_list[i]));
      // or svg.appendChild(this.getBoxSvg(i, data_list[i]));  // 簡易
    }
    $(jq_obj).append(svg);
  }
};
/*
Svg.prototype.getBoxSvg = function(i, d){
  if (ch_ck(d, "rect")){
    var rect = d["rect"];
    var elem = document.createElementNS(this.NS, "rect");
    elem.setAttribute("x", rect[0]);
    elem.setAttribute("y", rect[1]);
    elem.setAttribute("width", rect[2]);
    elem.setAttribute("height", rect[3]);
    elem.setAttribute("style","fill:red;opacity:0");
    return elem;
  }
};*/
Svg.prototype.getSvg = function(i, d){
  if (ch_ck(d, "type")){
    var type = d["type"];
    var offset = d["offset"];
    var rect = d["rect"];
    var vector = d["vector"];
    var elem = document.createElementNS(this.NS, d["type"]);
    if (type === "polygon"){
      elem.setAttribute("points", getSVGPoints(rect, vector, offset, null));
      elem.setAttribute("stroke","none");
      if (offset[1] === 0){
        elem.setAttribute("fill","green");
      }else{
        elem.setAttribute("fill","blue");
      }
      elem.setAttribute("id", d["ser"]);
    }
    $elem = $(elem);
    $elem.bind("mouseover", function(event){
      this.setAttribute("fill","red");
      wk_cv = event.currentTarget;
    });
    $elem.bind("mouseout", function(event){
```

```
      if (offset[1] === 0){
        this.setAttribute("fill","green");
      }else{
        this.setAttribute("fill","blue");
      }
    });
    return elem;
  }
  else{
    var element = document.createElementNS(this.NS, "rect");
    return element;
  }
};
/*
// Skeleton of Vein (for demonstration only)
Svg.prototype.createPlSvg = function(jq_obj, data, data_list){
  if (ch_ck(data)){
    var svg = document.createElementNS(this.NS, "svg");
    svg.setAttribute("width","100%");
    svg.setAttribute("height","100%");
    svg.setAttribute("viewBox","0 0 " + img_wid + " " + img_hei);
    for (var i = 0; i < data.length; i++){
      svg.appendChild(this.getSkSvg(i, data[i]));
    }
    if (ch_ck(data_list)){
      for (var i = 1; i < data_list.length; i++){
        svg.appendChild(this.getBoxSvg(i, data_list[i]));
      }
      $(jq_obj).append(svg);
    }
  }
};
Svg.prototype.getSkSvg = function(i, d){
  if (ch_ck(d, "type")){
    var type = d["type"];
    var data = d["data"];
    var elem = document.createElementNS(this.NS, "polyline");
    elem.setAttribute("points", getSVGSkPoints(data, type));
    elem.setAttribute("fill","none");
    elem.setAttribute("stroke-width","4");
    elem.setAttribute("stroke","blue");
    return elem;
  }
  else{
    var element = document.createElementNS(this.NS, "polyline");
    return element;
  }
};
*/
/*
Svg.prototype.getRectSvg = function(i, d){
  if (ch_ck(d, "type")){
    var type = d["type"];
    var data = d["data"];
    var elem = document.createElementNS(this.NS, "polygon");
    elem.setAttribute("points", getSVGRectPoints(data, type));
    elem.setAttribute("fill","none");
    elem.setAttribute("stroke-width","2");
    elem.setAttribute("stroke","green");
    return elem;
  }
  else{
    var element = document.createElementNS(this.NS, "polygon");
```

```
    return element;
  }
};*/

// Svg 関連関数
function getSVGPoints(d_rect, d_vector, d_offset, w_area){
  var p = getSVGPointsArray(d_rect, d_vector, d_offset, w_area);
  var svg_points = p[0][0] + "," + p[0][1];
  for (var j = 1; j < p.length; j++){
    svg_points = svg_points + " " + p[j][0] + "," + p[j][1];
  }
  return svg_points;
}
function getSVGPointsArray(d_rect, d_vector, d_offset, w_area){
  var p = [];
  var x_offset = d_offset[0];  // (d_rect[0] + d_rect[2]) - d_vector[0];  // 右上から文字の中心
  var y_offset = d_offset[1];  // d_vector[1] - d_rect[1];        // 上から文字の中心
  // RL-TB
  if (w_area === null){
    p[0] = [d_vector[0] - x_offset, d_vector[1] - y_offset];
    p[1] = [d_vector[0] + x_offset, d_vector[1] - y_offset];
    p[2] = [d_vector[0] + d_vector[2] + x_offset, d_vector[1] + d_vector[3] + y_offset];
    p[3] = [d_vector[0] + d_vector[2] - x_offset, d_vector[1] + d_vector[3] + y_offset];
  }
  else if (w_area.length === 3){
    if (w_area[2] === 1){
      w_area[2] = d_vector[0] + d_vector[2];
      w_area[3] = d_vector[1] + d_vector[3];
    }
    else {
      w_area[3] = w_area[1];
      w_area[2] = w_area[0];
      w_area[0] = d_vector[0];
      w_area[1] = d_vector[1];
    }
    if (w_area[1] < d_vector[1]) w_area[1] = d_vector[1];
    if (w_area[3] > d_vector[1] + d_vector[3]) w_area[3] = d_vector[1] + d_vector[3];

    dy1 = w_area[1] - d_vector[1];
    dx1 = d_vector[2] * dy1 / d_vector[3];
    dy3 = w_area[3] - d_vector[1];
    dx3 = d_vector[2] * dy3 / d_vector[3];

    w_vector = [d_vector[0] + dx1, d_vector[1] + dy1, dx3 - dx1, w_area[3] - w_area[1]];
    p[0] = [w_vector[0] - x_offset, w_vector[1] - y_offset];
    p[1] = [w_vector[0] + x_offset, w_vector[1] - y_offset];
    p[2] = [w_vector[0] + w_vector[2] + x_offset, w_vector[1] + w_vector[3] + y_offset];
    p[3] = [w_vector[0] + w_vector[2] - x_offset, w_vector[1] + w_vector[3] + y_offset];
  }
  else {
    // w_area = [x1, y1, x2, y2]
    if (w_area[3] > d_vector[1] + d_vector[3]) w_area[3] = d_vector[1] + d_vector[3];

    dy1 = w_area[1] - d_vector[1];
    dx1 = d_vector[2] * dy1 / d_vector[3];
    dy3 = w_area[3] - d_vector[1];
    dx3 = d_vector[2] * dy3 / d_vector[3];

    w_vector = [d_vector[0] + dx1, d_vector[1] + dy1, dx3 - dx1, w_area[3] - w_area[1]];
    p[0] = [w_vector[0] - x_offset, w_vector[1] - y_offset];
    p[1] = [w_vector[0] + x_offset, w_vector[1] - y_offset];
    p[2] = [w_vector[0] + w_vector[2] + x_offset, w_vector[1] + w_vector[3] + y_offset];
    p[3] = [w_vector[0] + w_vector[2] - x_offset, w_vector[1] + w_vector[3] + y_offset];
```

```
  }
  return p;
}
/*function getSVGSkPoints(d_lines, d_type){
  var p = getSVGSkPointsArray(d_lines, d_type);
  var svg_points = p[0][0] + "," + p[0][1];
  for (var j = 1; j < p.length - 0; j++){  // 最後を曲げる
    svg_points = svg_points + " " + p[j][0] + "," + p[j][1];
  }
  return svg_points;
}
function getSVGSkPointsArray(data, d_type){
  var p = [];
  var data1 = data.split(/[\s,]/);
  var ox = parseInt(data1[0]);
  var oy = parseInt(data1[1]);
  var ex = ox + parseInt(data1[2]);
  var ey = oy + parseInt(data1[3]);
  p[0] = [ox, oy];
  var j = 0;
  // RL-TB
  if (d_type === "text" ¦¦ d_type === "polyline"){
    for (j = 2; j < data1.length / 2; j++){
      p[j - 1] = [ox + parseInt(data1[j * 2]), oy + parseInt(data1[j * 2 + 1])];
    }
    j = j - 1;
  }
  else{
    p[1] = [ox + parseInt(data1[4]), oy + parseInt(data1[5])];
    p[2] = [ox + parseInt(data1[6]), oy + parseInt(data1[7])];
    j = 3;
  }
  p[j] = [ex, ey];  // 最後を曲げる
  return p;
}*/
/*
function getSVGRectPoints(d_lines, d_type){
  var p = getSVGRectPointsArray(d_lines, d_type);
  var svg_points = p[0][0] + "," + p[0][1];
  for (var j = 1; j < p.length; j++){
    svg_points = svg_points + " " + p[j][0] + "," + p[j][1];
  }
  return svg_points;
}
function getSVGRectPointsArray(data, d_type){
  var p = [];
  var data1 = data.split(/[\s,]/);
  var ox = parseInt(data1[0]);
  var oy = parseInt(data1[1]);
  var ex = ox + parseInt(data1[2]);
  var ey = oy + parseInt(data1[3]);
  p[0] = [ox, oy];
  p[1] = [ex, oy];
  p[2] = [ex, ey];
  p[3] = [ox, ey];
  return p;
}*/

// Vein の編集時 svg 作成用コンストラクタ
function Work(){
  this.NS = "http://www.w3.org/2000/svg";
  this.svg;
  this.g;
```

```
    this.spots = [];
    this.ox;
    this.oy;
    this.wx;
    this.wy;
    this.serial;
    this.last_serial;
    this.inclick = false;
    this.drag = false;
  }
  Work.prototype.initWork = function(jq_obj, data_list){
    this.svg = document.createElementNS(this.NS, "svg");
    this.svg.setAttribute("width", "100%");
    this.svg.setAttribute("height", "100%");
    this.svg.setAttribute("viewBox", "0 0 " + img_wid + " " + img_hei);
    this.svg.setAttribute("id", "work");
    this.g = document.createElementNS(this.NS, "g");
    var that = this;
    var endPoint = null;
    $(this.g).css({"cursor": "pointer"}).bind({
      "mousedown touchstart": function(evt){
        evt.stopPropagation();
        evt.preventDefault();
        that.inclick = true;
        endPoint = {'x': _pageX(evt), 'y': _pageY(evt)};
      },
      "mouseup touchend": function(evt){
        var endx = parseInt(endPoint.x);
        var endy = parseInt(endPoint.y);
        $("#tool").css({"left": addPx(endx + 16), "top": addPx(endy - 16)}).show();
      }
    });
    $(this.svg).append(this.g);
    $(jq_obj).append(this.svg);
  };
  Work.prototype.getWork = function(){
    // DOM
    var elem = document.createElementNS(this.NS, "polygon");
    elem.setAttribute("points", getSVGPoints([0, 0, 0, 0], [0, 0, 0, 0], [0, 0], null));
    elem.setAttribute("stroke", "none");
    elem.setAttribute("fill", "#0ff");
    // elem.setAttribute("class", "spot");
    this.g.appendChild(elem);
    $(elem).show();
    return $(elem);
  };
  Work.prototype.setStart = function(t_serial, tx, ty){
    this.clearWorks();
    $spot = this.getWork();
    this.spots.push($spot);
    $spot.attr("points", getSVGPoints(vein_list[t_serial]["rect"], vein_list[t_serial]["vector"],
vein_list[t_serial]["offset"], [tx, ty, tx + 1, ty + 1]));
    this.ox = tx;
    this.oy = ty;
    this.serial = t_serial;
    this.spots[0].show();
    this.last_serial = t_serial;
  };
  Work.prototype.setChange = function(t_serial, tx, ty){
    if (this.serial === null){
      this.clearWorks();
      $spot = this.getWork();
      this.spots.push($spot);
```

```
      $spot.attr("points", getSVGPoints(vein_list[t_serial]["rect"], vein_list[t_serial]["vector"],
vein_list[t_serial]["offset"], [tx, ty, tx + 1, ty + 1]));
      this.ox = tx;
      this.oy = ty;
      this.serial = t_serial;
      this.spots[0].show();
      this.last_serial = t_serial;
    } else if (t_serial > this.serial){
      if (this.last_serial > t_serial){
        this.clearWorks();
        this.spots.push(this.getWork());
      }
      var k = 0;
      this.spots[k].attr("points", getSVGPoints(vein_list[this.serial]["rect"], vein_list[this.serial]["vector"],
vein_list[this.serial]["offset"], [this.ox, this.oy, 1]));    // end fill
      for (var i = this.serial + 1; i < t_serial; i++){
        k = k + 1;
        if (this.spots[k] === undefined){
          this.spots.push(this.getWork());
        }
        this.spots[k].attr("points", getSVGPoints(vein_list[i]["rect"], vein_list[i]["vector"], vein_list[i]["offset"], null));
      }
      k = k + 1;
      if (this.spots[k] === undefined){
        this.spots.push(this.getWork());
      }
      this.spots[k].attr("points", getSVGPoints(vein_list[t_serial]["rect"], vein_list[t_serial]["vector"],
vein_list[t_serial]["offset"], [tx, ty, 0]));
      this.last_serial = t_serial;
    } else if (t_serial < this.serial){
      if (this.last_serial < t_serial){
        this.clearWorks();
        this.spots.push(this.getWork());
      }
      var k = 0;
      this.spots[k].attr("points", getSVGPoints(vein_list[this.serial]["rect"], vein_list[this.serial]["vector"],
vein_list[this.serial]["offset"], [this.ox, this.oy, 0]));
      for (var i = this.serial - 1; i > t_serial; i--){
        k = k + 1;
        if (this.spots[k] === undefined){
          this.spots.push(this.getWork());
        }
        this.spots[k].attr("points", getSVGPoints(vein_list[i]["rect"], vein_list[i]["vector"], vein_list[i]["offset"], null));
      }
      k = k + 1;
      if (this.spots[k] === undefined){
        this.spots.push(this.getWork());
      }
      this.spots[k].attr("points", getSVGPoints(vein_list[t_serial]["rect"], vein_list[t_serial]["vector"],
vein_list[t_serial]["offset"], [tx, ty, 1]));
      this.last_serial = t_serial;
    } else {
      if (this.last_serial !== t_serial){
        this.clearWorks();
        this.spots.push(this.getWork());
      }
      this.spots[0].attr("points", getSVGPoints(vein_list[this.serial]["rect"], vein_list[this.serial]["vector"],
vein_list[this.serial]["offset"], [this.ox, this.oy, tx, ty]));
      this.last_serial = t_serial;
    }
  };
  Work.prototype.setEnd = function(t_serial, tx, ty){
    this.drag = false;
```

```
  this.wx = tx;
  this.wy = ty;
};
Work.prototype.clearWorks = function(){
  $("#wk_base polygon").remove();
  for (var i= this.spots.length - 1; i >= 0; i--){
    var spot = this.spots.pop();
    spot = null;
  }
};
Work.prototype.getRect = function(){
  var d = this.spots[0].attr("points").split(/[\s,]/);
  var area = [parseInt(d[0]), parseInt(d[1]), parseInt(d[0]), parseInt(d[1])];
  for (var i = 2; i < d.length; i = i + 2){
    if (parseInt(d[i]) < area[0]) area[0] = parseInt(d[i]);
    if (parseInt(d[i + 1]) < area[1]) area[1] = parseInt(d[i + 1]);
    if (parseInt(d[i]) > area[2]) area[2] = parseInt(d[i]);
    if (parseInt(d[i + 1]) > area[3]) area[3] = parseInt(d[i + 1]);
  }
  for (var j = 1; j < this.spots.length; j++){
    d = this.spots[j].attr("points").split(/[\s,]/);
    for (var i = 0; i < d.length; i = i + 2){
      if (parseInt(d[i]) < area[0]) area[0] = parseInt(d[i]);
      if (parseInt(d[i + 1]) < area[1]) area[1] = parseInt(d[i + 1]);
      if (parseInt(d[i]) > area[2]) area[2] = parseInt(d[i]);
      if (parseInt(d[i + 1]) > area[3]) area[3] = parseInt(d[i + 1]);
    }
  }
  return [area[0], area[1], area[2] - area[0], area[3] - area[1]];
};
Work.prototype.getArea = function(){
  var area = this.getRect();
  return area[2] * area[3];
};

// Vein の Application レイヤ
function App(){
  this.NS = "http://www.w3.org/2000/svg";
  this.svg;
  this.inclick = false;
}
App.prototype.initApp=function(jq_obj){
  this.svg = document.createElementNS(this.NS, "svg");
  this.svg.setAttribute("width", "100%");
  this.svg.setAttribute("height", "100%");
  this.svg.setAttribute("viewBox", "0 0 " + img_wid + " " + img_hei);
  this.svg.setAttribute("id", "ap_svg");
  $(jq_obj).append(this.svg);
};

// Vein のハイライト・コンストラクタ
function Spot(jq_obj, t_id, t_value){
  this.NS = "http://www.w3.org/2000/svg";
  this.g;
  this.spots = [];
  this.id = t_id;

  //SVG
  this.g = document.createElementNS(this.NS, "g");
  //this.g.setAttribute("width", "100%");
  //this.g.setAttribute("height", "100%");
  //this.g.setAttribute("viewBox", "0 0 "+img_wid+" "+img_hei);
  this.g.setAttribute("id", this.id);
```

```
    var that = this;
    var endPoint = null;
    $(this.g).bind({
      "mousedown touchstart": function(evt){
        evt.stopPropagation();
        evt.preventDefault();
        ap_layer.inclick = true;
        endPoint = {'x': _pageX(evt), 'y': _pageY(evt)};
        ap_cv = {"id": $(this).attr("id"), "spot": $(this)};
      },
      "mouseup touchend": function(evt){
        activeVeinTool(endPoint);
        $("polygon",this).css({"cursor":"pointer"}).attr({"fill":"#90f"});
      },
      "mouseover": function(evt){
        $("polygon",this).css({"cursor":"pointer"}).attr({"fill":"#90f"});
      },
      "mouseout": function(evt){
        if (ap_cv == null){
          $("polygon",this).css({"cursor":"default"}).attr({"fill":"#09f"});
        }
      }
    });
    $(jq_obj).append(this.g);
  }
  Spot.prototype.setSpot=function(array){
    var obj = [];
    if (parseInt(array[1].t_serial) === parseInt(array[0].t_serial)){
      if (array[1].t_y >= array[0].t_y){
        obj[0] = array[0];
        obj[1] = array[1];
      } else {
        obj[0] = array[1];
        obj[1] = array[0];
      }
      this.spots.push(this.getSpot());
      this.spots[0].attr("points", getSVGPoints(vein_list[obj[0].t_serial]["rect"], vein_list[obj[0].t_serial]["vector"],
vein_list[obj[0].t_serial]["offset"], [obj[0].t_x,obj[0].t_y,obj[1].t_x,obj[1].t_y]));
    }
    else {
      if (parseInt(array[1].t_serial) > parseInt(array[0].t_serial)){
        obj[0] = array[0];
        obj[1] = array[1];
      } else {
        obj[0] = array[1];
        obj[1] = array[0];
      }

      var k = 0;
      this.spots.push(this.getSpot());
      this.spots[k].attr("points", getSVGPoints(vein_list[obj[0].t_serial]["rect"], vein_list[obj[0].t_serial]["vector"],
vein_list[obj[0].t_serial]["offset"], [obj[0].t_x,obj[0].t_y,1]));
      for (var i = parseInt(obj[0].t_serial) + 1; i < obj[1].t_serial; i++){
        k = k + 1;
        this.spots.push(this.getSpot());
        this.spots[k].attr("points", getSVGPoints(vein_list[i]["rect"], vein_list[i]["vector"], vein_list[i]["offset"],null));
      }
      k = k + 1;
      this.spots.push(this.getSpot());
      this.spots[k].attr("points", getSVGPoints(vein_list[obj[1].t_serial]["rect"], vein_list[obj[1].t_serial]["vector"],
vein_list[obj[1].t_serial]["offset"], [obj[1].t_x,obj[1].t_y,0]));
    }
  };
```

```
Spot.prototype.getSpot=function(){
//DOM
  var elem = document.createElementNS(this.NS, "polygon");
  elem.setAttribute("points", getSVGPoints([0, 0, 0, 0], [0, 0, 0, 0], [0, 0], null));
  elem.setAttribute("stroke", "none");
  elem.setAttribute("fill", "#09f");
  //elem.setAttribute("id", "");
  this.g.appendChild(elem);
  $(elem).show();
  return $(elem);
};
Spot.prototype.clearSpots = function(){
  for (var i = this.spots.length - 1; i >= 0; i--){
    var spot = this.spots.pop();
    spot = null;
  }
};

// 初期起動
Vein.init = function(a_selecter){

  // 実装の一例
  if (a_selecter == null) a_selecter = "#view #img";  // 階層
  var selecters = a_selecter.split(/\s/);
  $view = $(selecters[0]);    // View
  $img = $(a_selecter);       // Image target

  var vein_name = $img.attr("data-vein");
  // 仮に，上記の記述が存在する場合のみを対象とする
  if (ch_ck(vein_name)){

    img_wid = $img.width();
    img_hei = $img.height();


    $.getJSON("../json/"+vein_name + ".json", function(json){

      vein_list = Util.getVeinList(json);

      // Vein 用レイヤ組み込み
      $sk_base = $('<div id="sk_base"></div>');    // Skeleton Layer: SVG による TextVein 確認レイヤ，非表示，非実行
      $ap_base = $('<div id="ap_base"></div>');    // Application (Spot) Layer: 保存結果の（Spot）表示レイヤ
      $wk_base = $('<div id="wk_base"></div>');    // Work Layer: 選択機能の実装
      $vein_base = $('<div id="vein_base"></div>');    // Vein Layer: SVG による Vein レイヤ（最上位）
      $view.append($sk_base).append($ap_base).append($wk_base).append($vein_base);

      //sk_layer = new Svg();
      //sk_layer.createSkSvg($("#sk_base"), json, vein_list);

      wk_layer = new Work();
      wk_layer.initWork($("#wk_base"), vein_list);

      vein_layer = new Svg();
      vein_layer.createSvg($("#vein_base"), vein_list);

      ap_layer = new App();
      ap_layer.initApp($("#ap_base"));

      var vein_id;

      var vein_tool_active = false;
      var vein_tool_active_timerid;
      var VEIN_TOOL_POPUP_TIME = Vein.waketime;
```

```
var endPoint = null;
var inedit = false;
$("#vein_base").bind({
  "mousedown touchstart": function(evt){
    evt.stopPropagation();
    evt.preventDefault();
    Util.checkInOut(vein_list, _elemX(evt, $(this).offset().left) / vscale, _elemY(evt, $(this).offset().top) / vscale);
    endPoint = {'x': _elemX(evt, $(this).offset().left), 'y': _elemY(evt, $(this).offset().top)};
    if (wk_cv === null){
    } else {
      //evt.stopPropagation();
      //evt.preventDefault();
      // SVG に含めることができるのは id 属性
      vein_serial = parseInt(wk_cv.ser.substring(1));
      wk_layer.setStart(vein_serial, _elemX(evt, $(this).offset().left) / vscale, _elemY(evt, $(this).offset().top) / vscale);
    }
    inedit = true;
    vein_tool_active_timerid = setTimeout(function(){
      var endx = _pageX(evt);
      var endy = _pageY(evt);
      $("#vein_tool").css({"left": addPx(endx + 16), "top": addPx(endy - 16)}).show();
      $("#wk_base polygon").attr({"fill": "#0f0"});
      vein_tool_active = true;
    }, VEIN_TOOL_POPUP_TIME);
  },
  "mousemove touchmove": function(evt){
    ap_layer_active = false;
    Util.checkInOut(vein_list, _elemX(evt, $(this).offset().left) / vscale, _elemY(evt, $(this).offset().top) / vscale);
    if (wk_cv === null){
    } else {
      if (inedit){
        vein_serial = parseInt(wk_cv.ser.substring(1));
        wk_layer.setChange(vein_serial, _elemX(evt, $(this).offset().left) / vscale, _elemY(evt, $(this).offset().top) / vscale);
        endPoint = {'x': _elemX(evt, $(this).offset().left), 'y': _elemY(evt, $(this).offset().top)};
      }
    }
  },
  "mouseup touchend": function(evt){
    ap_layer_active = false;
    clearTimeout(vein_tool_active_timerid);
    inedit = false;
    //Util.checkInOut(vein_list, _elemX(evt, $(this).offset().left) / vscale, _elemY(evt, $(this).offset().top) / vscale);
    if (wk_cv === null){
      // No work
    } else {
      vein_serial = parseInt(wk_cv.ser.substring(1));
      var endx = endPoint.x;
      var endy = endPoint.y;
      wk_layer.setEnd(vein_serial, parseInt(endx / vscale), parseInt(endy / vscale));
    }
    if (vein_tool_active){
      $(this).hide();
      vein_tool_active = false;
    }
  }
});

// 無効化
$("#wk_base").bind({
  "mousedown touchstart": function(evt){
    return false;
  },
  "mouseup touchend": function(evt){
```

```
          if(!wk_layer.inclick){
            //$("#note").hide();
            wk_layer.clearWorks();
            $vein_base.show();
          } else {
            wk_layer.inclick = false;
            $vein_base.show();
          }
          $("#vein_tool").hide();
        }
      });

      $("#vein_tool").hide();
      $("#vein_menu").css({"opacity": "0.5"}).show();

      $("#sk_base").hide();
      $("#ap_base").hide();
      $("#wk_base").hide();
      $("#vein_base").hide();

      showVeins();
    });
  }
};


var activeVeinTool = function(a_endpoint){
  if (a_endpoint != null){
    var endx = parseInt(a_endpoint.x);
    var endy = parseInt(a_endpoint.y);
    $("#vein_tool").css({"left": addPx(endx + 16), "top": addPx(endy - 16)}).show();
  }
};

// 参考実装
var saveVein = function(){
  arguments.length > 0 ? type = arguments[0] : type = "spot";
  var vdata = Vein.veinobj[type];
  vdata["type"] = "spot";    // 登録用 Vein の種類を 'spot' と定義（拡張）
  vdata["url"] = window.location.href;
  vdata["image"] = $("#img").attr("src");
  vdata["veindata"] = $("#img").attr("data-vein");
  //   ローカルストレージへの保存
  var now = new Date();
  vdata["createdate"] = now.getTime();
  var header = vdata["veindata"].replace(/\//, "_") + "__";
  var vid = vdata.p1 + "_" + vdata.p1x + "_" + vdata.p1y + "_" + vdata.p2 + "_" + vdata.p2x + "_" + vdata.p2y;
  vdata["vid"] = vid;
  localStorage.setItem(header+vid, JSON.stringify(vdata));
  showVeins();
};
var showVeins = function(){
  arguments.length > 0 ? type = arguments[0] : type = "spot";
  var sorts = [];
  for(var ky in localStorage){
    var header = $("#img").attr("data-vein").replace(/\//, "_")+"__";
    if (ky.indexOf(header) < 0) continue;
    var val = localStorage[ky];
    var value = JSON.parse(val);  //var spot = eval("("+val+")");
    if (value["type"] === type){
      sorts.push(value);
    }
  }
```

```
  sorts.sort(function(a, b){
    return a["areaval"] - b["areaval"];
  });
  for(var k = sorts.length - 1; k >= 0; k--){
    var value = sorts[k];
    var idkey = value["veindata"].replace(/\//, "_") + "__" + value["vid"];
    if (typeof apps[idkey] === "undefined"){
      activeSpots(idkey, value);
    }
  }
};
var removeVein = function(idkey){
  var tid = "#" + idkey;
  if ($(tid).length > 0){
    $(tid).remove();
    var obj = apps[idkey];
    obj.clearSpots();
    obj = null;
    delete localStorage[idkey];
    showVeins();
  }
};
var activeSpots = function(idkey, value){
  if ($("#" + idkey).length < 1){
    var spot = new Spot($("#ap_svg"), idkey);
    var index1 = value.p1.indexOf("_p");
    var index2 = value.p2.indexOf("_p");
    var obj1 = {"t_serial": value.p1.substring(index1 + 2), "t_id": value.p1.substring(0, index1), "t_x": value.p1x, "t_y": value.p1y};
    var obj2 = {"t_serial": value.p2.substring(index2 + 2), "t_id": value.p2.substring(0, index2), "t_x": value.p2x, "t_y": value.p2y};
    spot.setSpot([obj1, obj2]);
    apps[idkey] = spot;
  }
};

// 登録
Vein.UI.entry = function(){
  if (vein_list[wk_layer.serial] == null) return;
  var hl = {};
  hl["p1"] = vein_list[wk_layer.serial]["id"] + "_p" + wk_layer.serial;
  hl["p1x"] = Math.floor(wk_layer.ox);
  hl["p1y"] = Math.floor(wk_layer.oy);
  hl["p2"] = vein_list[wk_layer.last_serial]["id"] + "_p" + wk_layer.last_serial;
  hl["p2x"] = Math.floor(wk_layer.wx);
  hl["p2y"] = Math.floor(wk_layer.wy);
  hl["maxrect"] = wk_layer.getRect();
  hl["areaval"] = wk_layer.getArea();
  Vein.veinobj["spot"] = hl;
  saveVein("spot");
  Vein.UI.cancel();
};

// 登録時キャンセル操作
Vein.UI.cancel = function(){
  $("#vein_tool").hide();
  if (Vein.inapp){
    if (ap_cv != null) $("polygon", ap_cv["spot"]).attr({"fill": "#09f"});
    ap_cv = null;
  }
  else{
    wk_layer.clearWorks();
    $("#wk_base").show();
    $("#vein_base").show();
  }
```

```
  return false;
};

// 登録済付箋（Spot）の削除
Vein.UI.remove = function(){
  if (!Vein.inapp){
    // 登録時はキャンセルに同じ
  //} else if (confirm(" 登録データを削除しますか？ ")){
  }
  else{
    removeVein(ap_cv["id"]);
  }
  Vein.UI.cancel();
};

// 参考
Vein.UI.deleteMem = function(){
  if (confirm(" この画像の登録済み付箋を削除しますか？ ")){
    //deleteVeins();
  }
};
Vein.UI.deleteMemAll = function(){
  if (confirm(" すべての登録済み付箋を削除しますか？ ")){
    //deleteAllVeins();
  }
};

// メニュー
Vein.UI.hideVein = function(){
  $("#vein_tool").hide();
  $("#sk_base").hide();
  $("#vein_base").hide();
  $("#ap_base").hide();
  $("#wk_base").hide();
  $("#vein_menu").css({"opacity": "0.5"});
  Vein.inapp = false;
  Vein.inuse = false;
};
Vein.UI.useVein = function(){
  $("#vein_tool").hide();
  $("#sk_base").hide();
  $("#vein_base").show().focus();
  $("#ap_base").show();
  $("#wk_base").show();
  $("#vein_menu").css({"opacity": "1"});
  $("#vein_tool_entry").show();
  $("#vein_tool_remove").hide();
  Vein.inapp = false;
  Vein.inuse = true;
  showVeins();
};
Vein.UI.useApp = function(){
  $("#vein_tool").hide();
  $("#sk_base").hide();
  $("#vein_base").hide();
  $("#ap_base").show();
  $("#wk_base").hide();
  $("#vein_menu").css({"opacity": "1"});
  $("#vein_tool_entry").hide();
  $("#vein_tool_remove").show();
  Vein.inapp = true;
  Vein.inuse = false;
  showVeins();
```

```
  };

}(window, window.jQuery));
```

B. コアプログラム用スタイルシート（CSS）

```
@charset "utf-8";
/* Vein CSS */
/* 基礎 */
body{
  margin: 0;
  padding: 0;
  -webkit-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
}
#view{
  position: absolute;
}
#image{
  position: absolute;
  left: 0;
  top: 0;
}
/* Vein Layer */
#sk_base{
  /* Skeleton */
  position: absolute;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  opacity: 1.0;
}
#vein_base{
  position: absolute;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  opacity: 0.0;
}
#wk_base{
  position: absolute;
  left:0;
  top:0;
  width: 100%;
  height: 100%;
  opacity: 0.3;
}
#wk_base div{
  position: absolute;
}
#ap_base{
  position: absolute;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
```

```
  opacity: 1;
}
#ap_base g{
  opacity: 0.2;
}
/* Vein メニュー（略）*/
/* 編集メニュー（略）*/
```

付記

・HTML（HTML5）および画像のビューア関係の JavaScript と CSS，画像は除いている。
・動作には，jQuery ライブラリが必要。作成時のバージョンは 1.9.1 を用いている。
・全体のコードおよびその動作サンプルについては，http://www.ipallet.org/veins-sample/index.html からリンクを参照のこと（認証用ユーザー名：guest，パスワード：fusen）出稿後に判明した問題，以降のバージョン情報，テキスト動線の製作アプリケーションなどについての情報も，ここに記録している。
・掲載のコアプログラムでは，本稿の主旨上必須でない機能は除いている。
・検証は，Windows 7 Service Pack 1 上の HTML5 対応ブラウザ，および Mac OS X 10.8 上の HTML5 対応ブラウザ，および iOS 6 デバイスで行なった。各製品の商標記載は省略する。
・コードにはコメント化した参考箇所を含めて記載している。変数名やコード，コメントの整理は十分でないことは認識しているがそのまま掲載する。